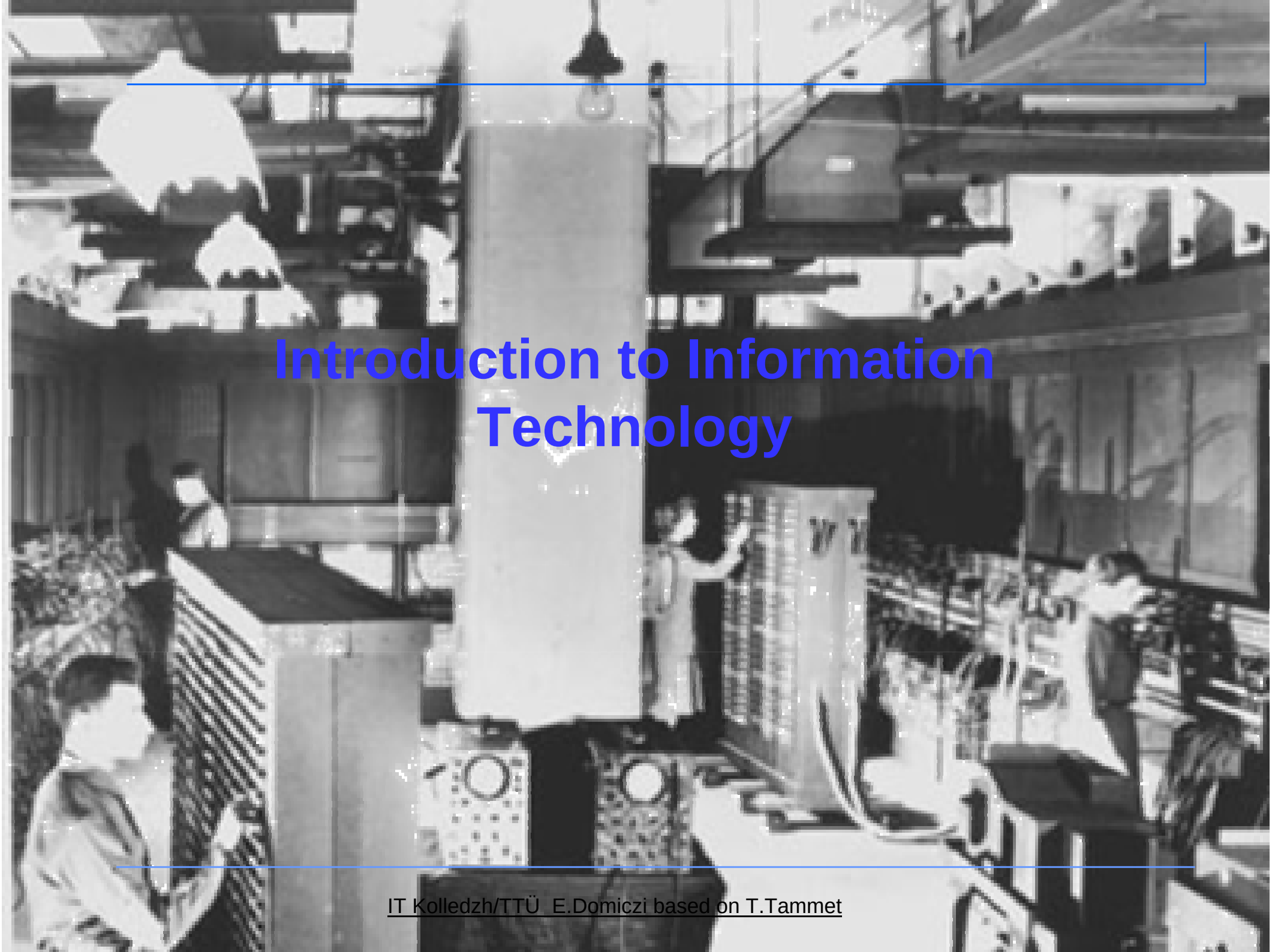


Introduction to Information Technology

Software architecture and paradigms

Components of Software packages: 1

- In principle software is built of many components. These components can be classified as follows:

- Complete end-user applications (word-processing, bookkeeping, chess, etc.), which often (not always) can be connected/integrated into other SW's via programming.

For instance Microsoft Office applications (Word, Excel, etc) can be bound via the programmable VisualBasic development package. One has to take into account that these programs have to be bought and installed on the computer.

This scheme is more often used on the Windows OpSystem. Many database systems are analogous, e.g. Oracle, Sybase, etc. Whereby sometimes the one database system can be changed to another similar one.

Components of Software packages: 2

- Big ready-components (database-server, www-server, mail-server components, graphic servers (like X11), OpSystems themselves, etc.

These programs can be used stand-alone, but typically are customized for the end-users as special separate applications, a la:

```
.....
cursor=con.cursor()
sqlstr=""”select clientid, clientname
              from clientbase
              where clientname like ‘%Jaan%’”””
cur.execute(sqlstr)
results=cur.fetchall()
for i in results
    print “id: “,i[0],”name: “,i[1]
 ......
```

Components of Software packages: 3

- Concrete, bounded functions (graphical drawing to screen, arithmetics, file reading and writing, build up an internet connection to a certain address, etc.) are realised by small components and their sets, so-called libraries.

- **For the distribution of such components a set of compilers and other SW-development tools are used and they can be used only with those programming language and development environment setup.**

Closed and open systems

- Network of open and closed systems.
  - Many SW components serve many users in server computers, which are connected to a network.
    For private users they basically offer siervices. Primary is the easy connectivity into a network and with other software components.
  - For such an approach the main basic platform is a Unix-based server, with big databases and search-applications.
    The end user pc communicates through net (mainly internet) with the server and can have any operating system.
    Often the user communicates via an ordinary Browser, which can execute Java-language programs.

Libraries

- View 3: application libraries

- Most languages do NOT have standard libraries or they are very limited
  - E.g.: ANSI C. ANSI C ***library*** contains string-processing, file-processing, printing, and a couple of analog categories.

- As a rule there are powerful application-libraries (graphics, windows, mouse-handling, network connections, parallel processing, etc…)

  non-standard, limited HardWare with OpSystem
  (Windows 95, Windows NT, X windows, Linux, ...)

- Almost the only exception: Java

Practical application domains

- Universal, except for applications
  - requiring high speed or
  - for operating systems:

  Java

- Applicatioins requiring maximum speed,
  UNIX system-programming: C, C++
- Windows application programming: VisualBasic, Java, Delphi, C
- Network-client programming: Java + Javascript
- Server programming: Java, Perl, PHP, Python, C, ..
- Specific applications: as per requirements

Development environments

- Programming languages, compiler
  - As a rule to create SW one has to be able to write in a specific language, which is possible via special education and long practical experience.
  - In the world there are thousands of programming languages, more widespread are about ten languages.
  - Programs written in a specific programming language are transformed into a program executable on a computer by a so-called compiler.

Development environments

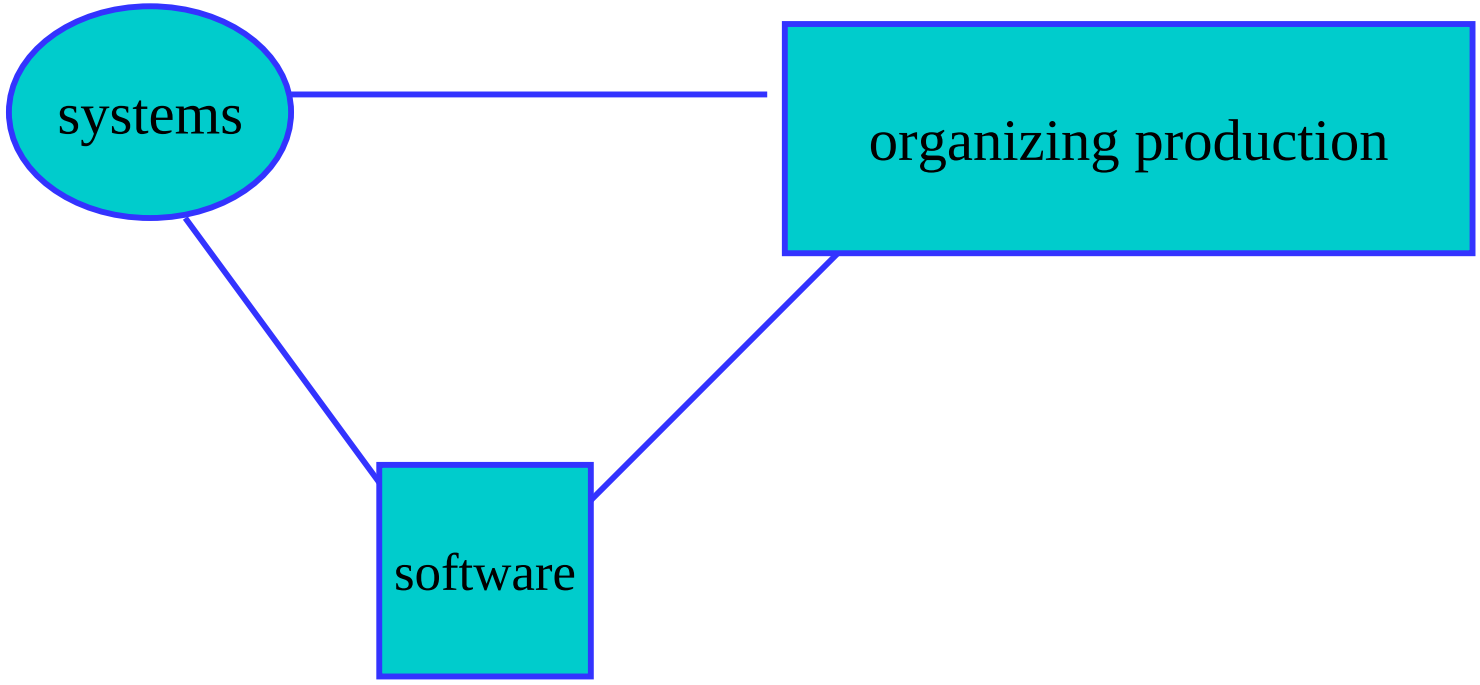
- visual development environments
  - To make programmers' work easier, so-called visual development environments have been created. Thanks to them in simpler cases one can reduce the work amount and time to create a program.
  - These are especially suited to create data-input forms, based on the input form they create the needed program automatically.
  - In case of non-trivial problems it's indispensable to combine visual drawing-methods with classical programming.

Development environments

- module, plug-in
  - The majority of modern programming languages are layered, containing a small and relatively simple core and a big amount of small SW components (so-called “modules”, “plug-ins”) for several standard tasks.

Development of IT system paradigms

Historical views:



architectural phases

1945-1970 1970-1995 1995-2020?

A) Systems' architectural phases

Mainframes - Microcomputers - Network systems

B) SoftWare Platform architectural phases:

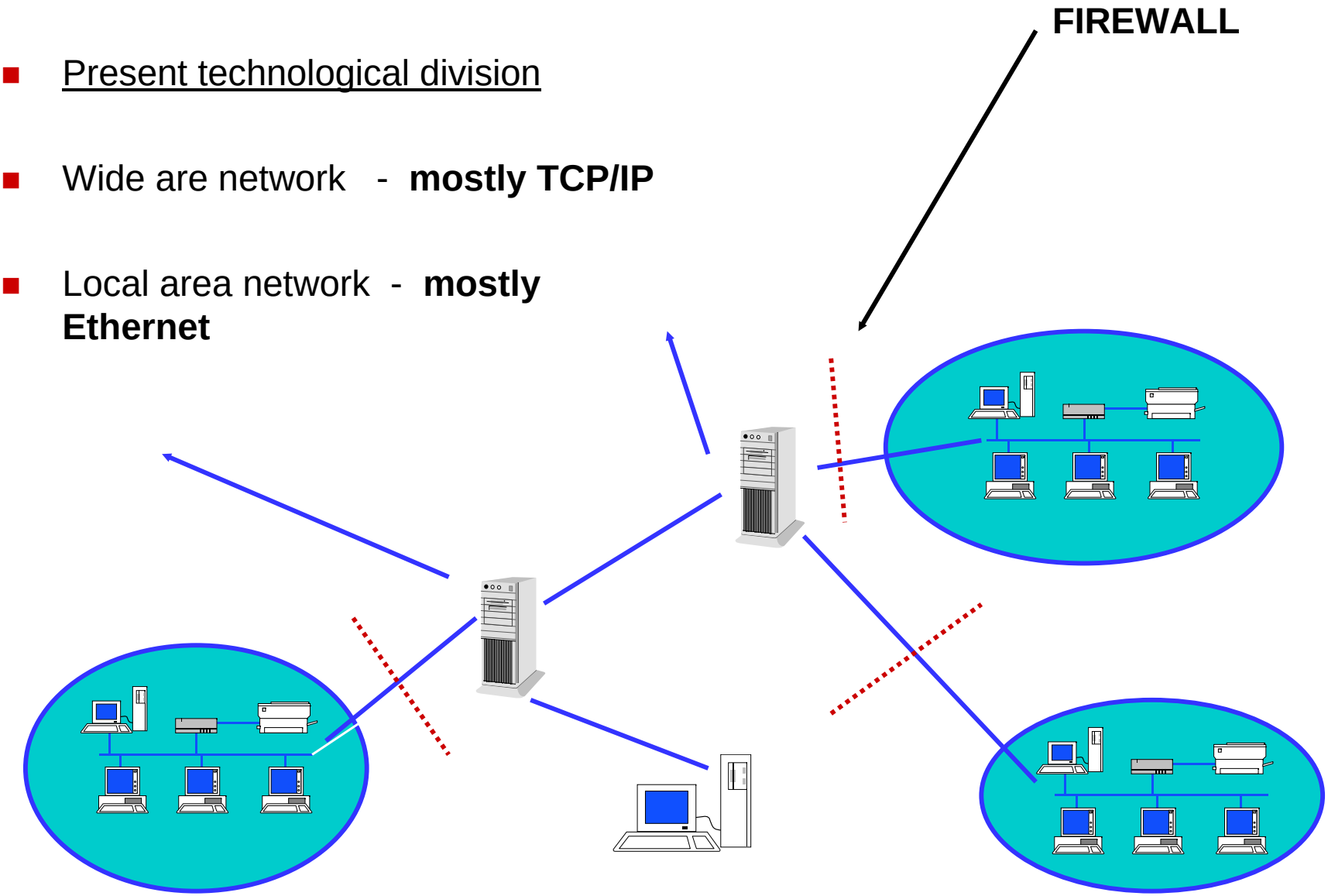
assembler, pure languages -
***libraries*, development environments, components -**
component binding

C) Organizing production architectural phases:

big company, open - small company, closed -
free components, binding, maintenance

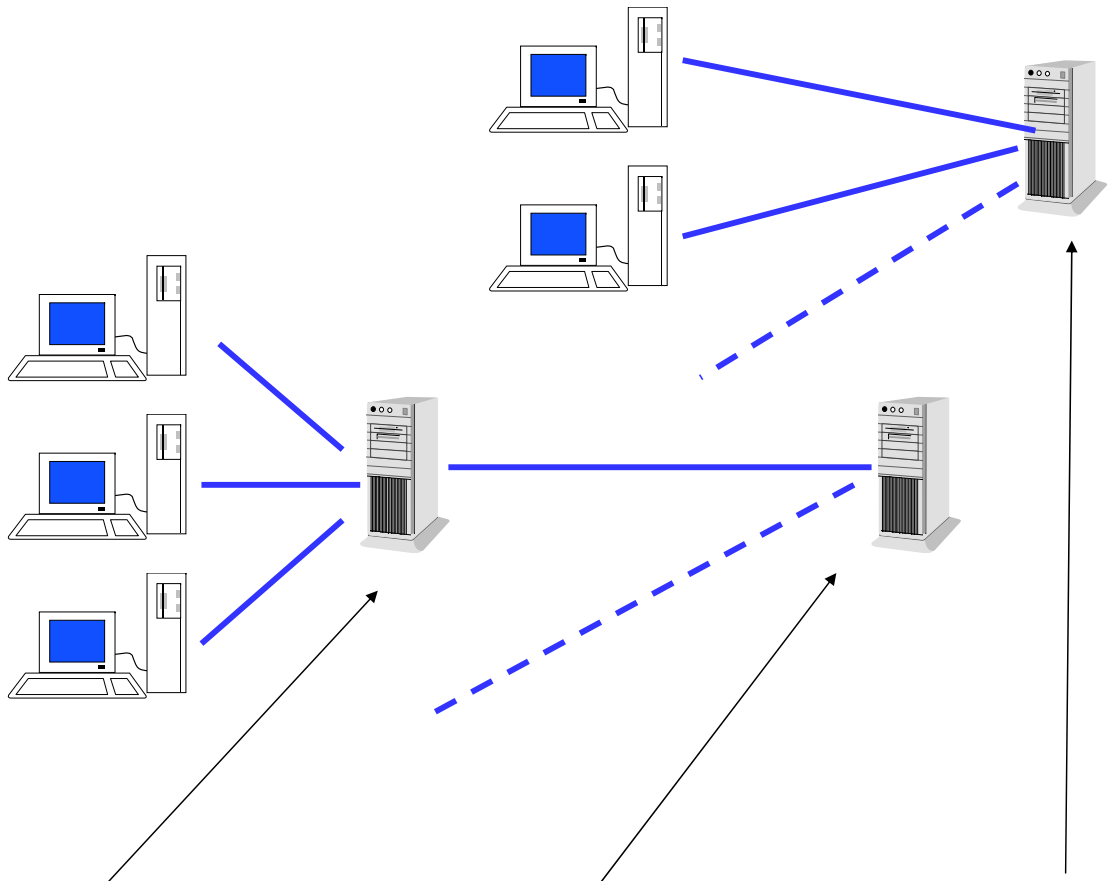
Network applications: basic terminology

- Present technological division
- Wide are network - **mostly TCP/IP**
- Local area network - **mostly Ethernet**



Network applications: client-server architectures

- 2-layer
- 3-layer
- N-layer



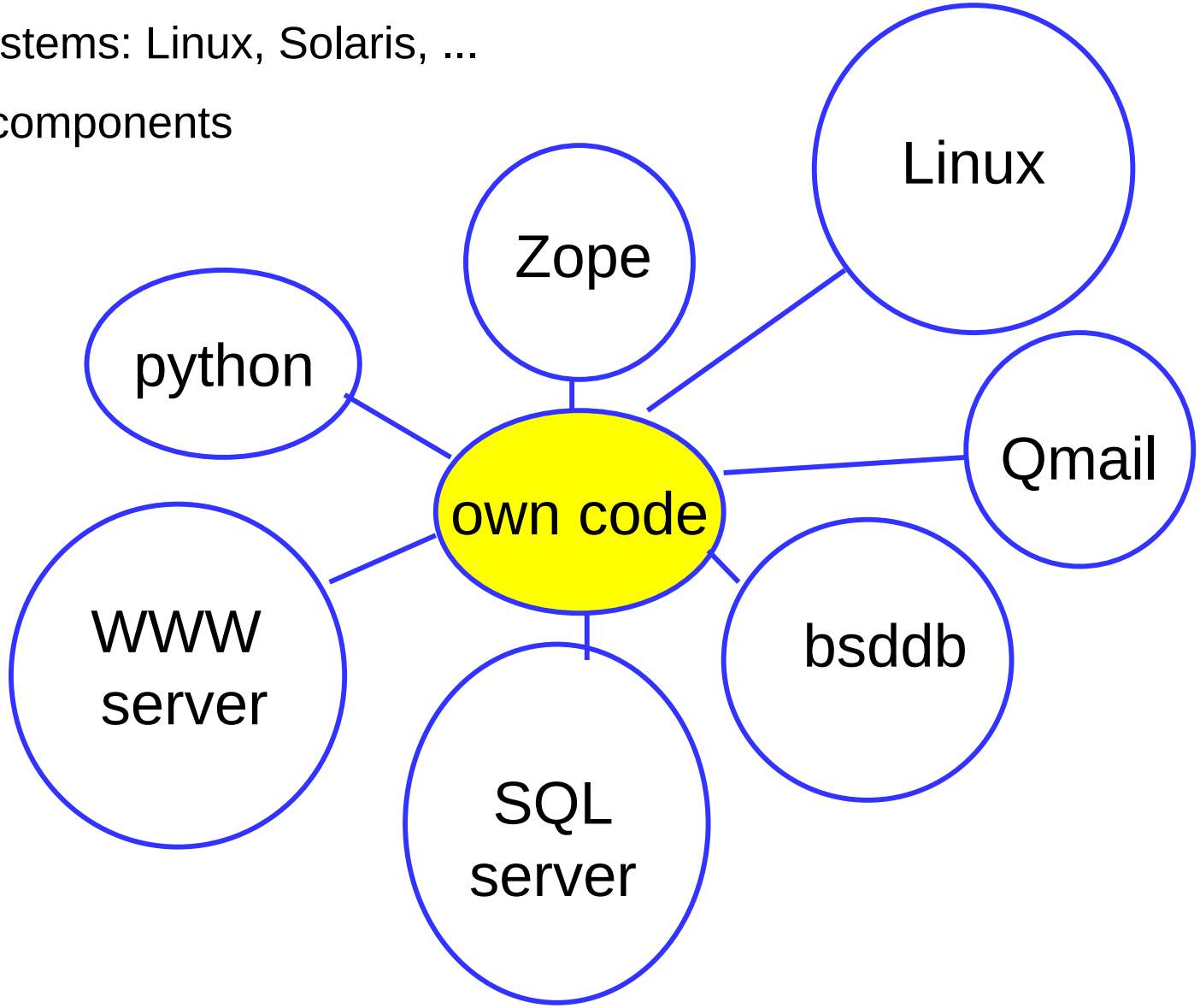
Few connected clients, adjustment distributed

Does part of the work, doesn't need constant adjustment

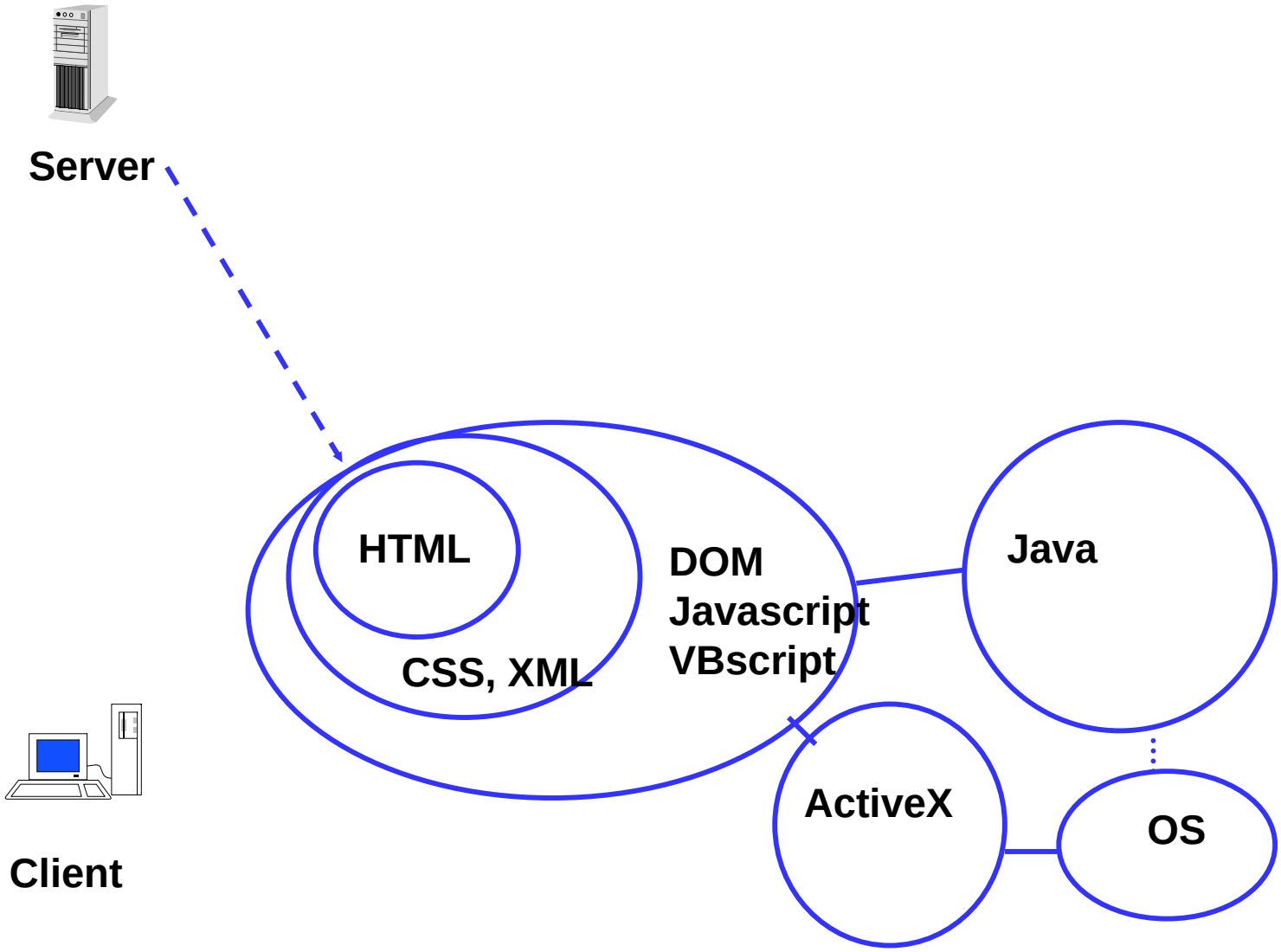
Does all the work, needs constant adjustment

Example: server-side technology of one system

- Operating systems: Linux, Solaris, ...
- Free, open components



Network software current architecture



Network software current architecture

Protocols

- Basic protocols
  - IP
  - TCP
  - HTTP
  - CGI
- Layout protocols
  - HTML - hypertext markup language
  - CSS - cascading style sheets
  - DOM - document object model
  - XML - extended markup language
- Further developed, distributed object protocols
  - RMI
  - CORBA
  - DCOM
  - JINI
  - XML-RPC
  - SOAP applications

Network software current architecture

Application platforms

- Server side
  - Script languages together with libraries: Perl, Python, Javascript, VBScript
  - Servlets: Java
  - Databases
  - Application servers
- Client side
  - Scripting languages: Javascript, VBScript,
  - Applets, etc. Java, Flash
  - Applications: Java, ActiveX

Network applications: why?

A) Needs of individual user:

- Accessing own documents and materials and programs anytime, anywhere:
  - documents, notes
  - calendar
  - contact database
  - bank account
  - e-mail
  - familiar software

Network applications: why?

B) Groupwork and management needs:

- Distributed groupwork : access to the common documents and materials of the group anythime, anywhere
  - common calendar, time planning, group discussion in real time
  - documents and databases the group is working on

- Access to documents and databases based on access rights anytime, anywhere:
  - bookkeeping
  - company's bank account
  - company's databases
  - workers' reports and documents

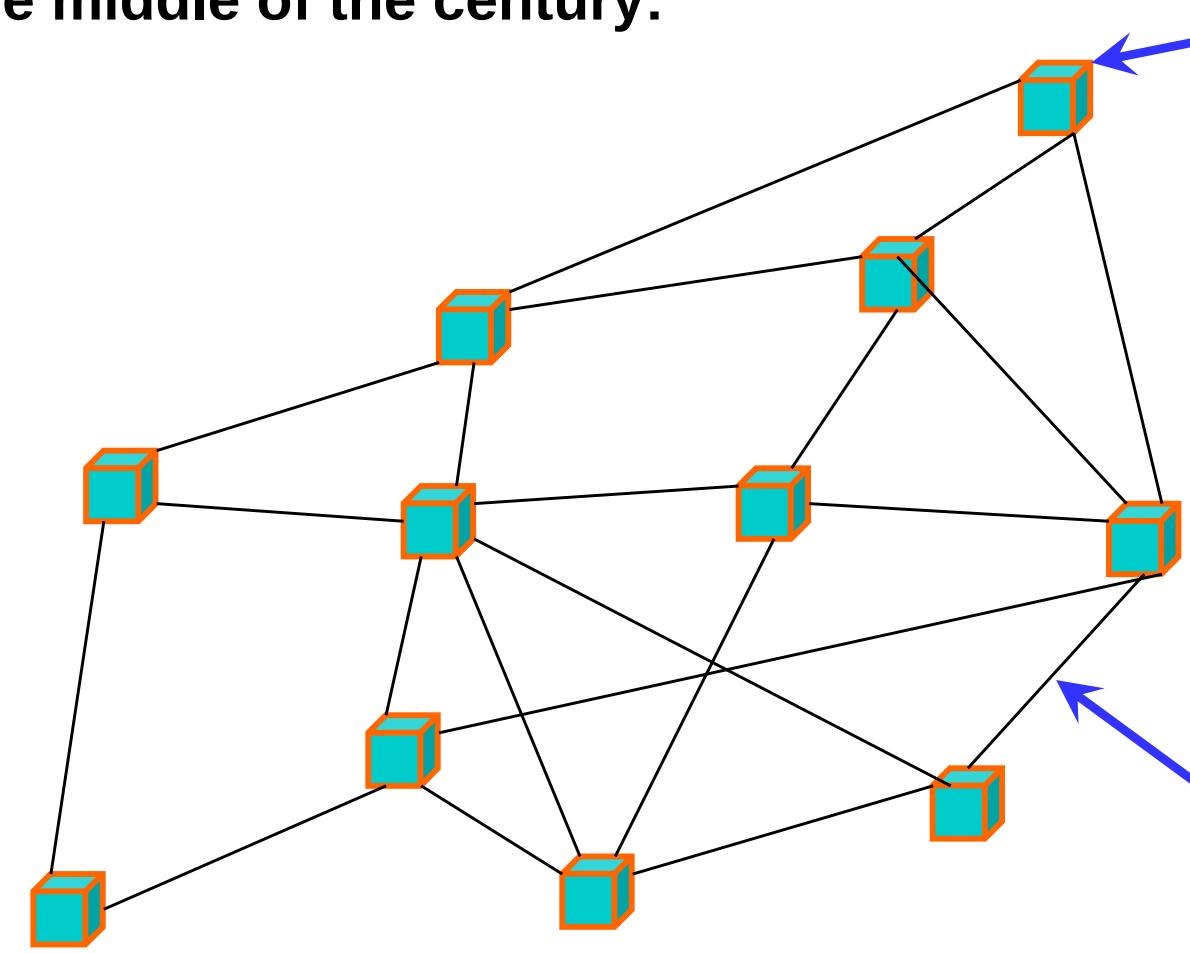
Network applications: why?

C) Automatic data storage and retrieval:

- **Automatic archiving and enabling convenient archive searching**
- **Erinevates osakondades ja ettevõtetes olevate eriostarbeliste andmebaaside andmete ühendamine, ühispäringute sooritamine**
- **(Semi-)Automatic Info search from the network**

Intelligent network !?

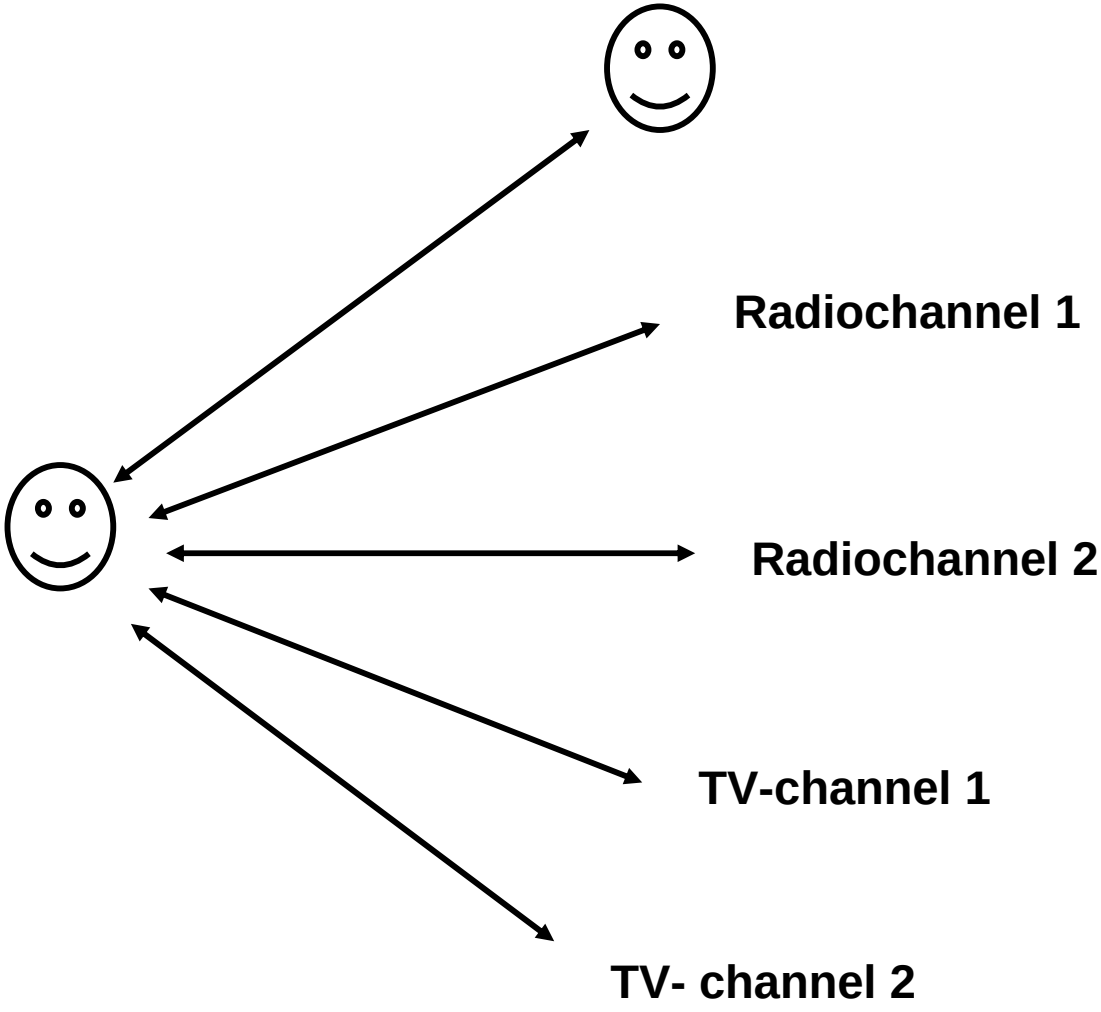
In the middle of the century:



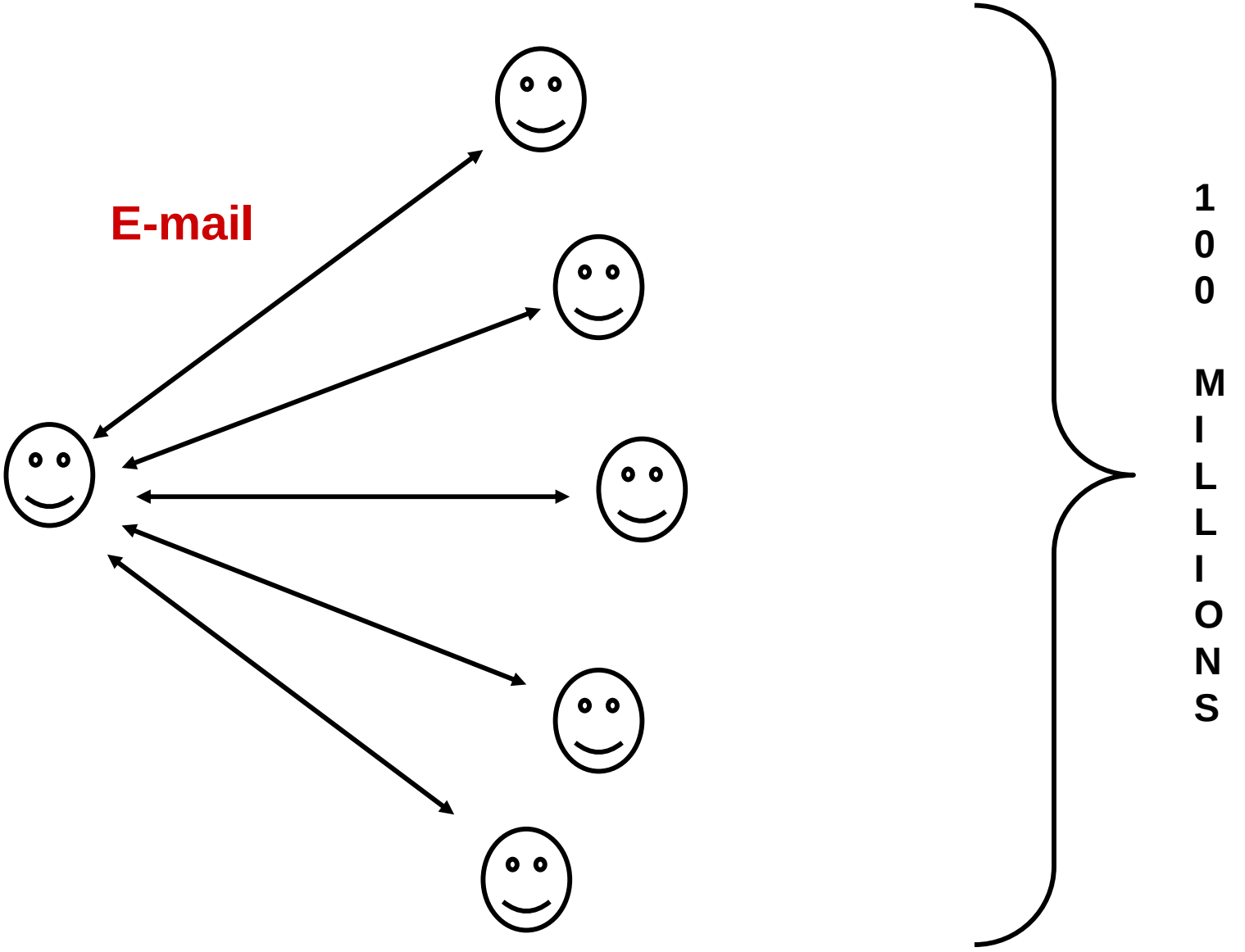
Every computer is 1000 times more powerful

The network has 1000 times more information and services

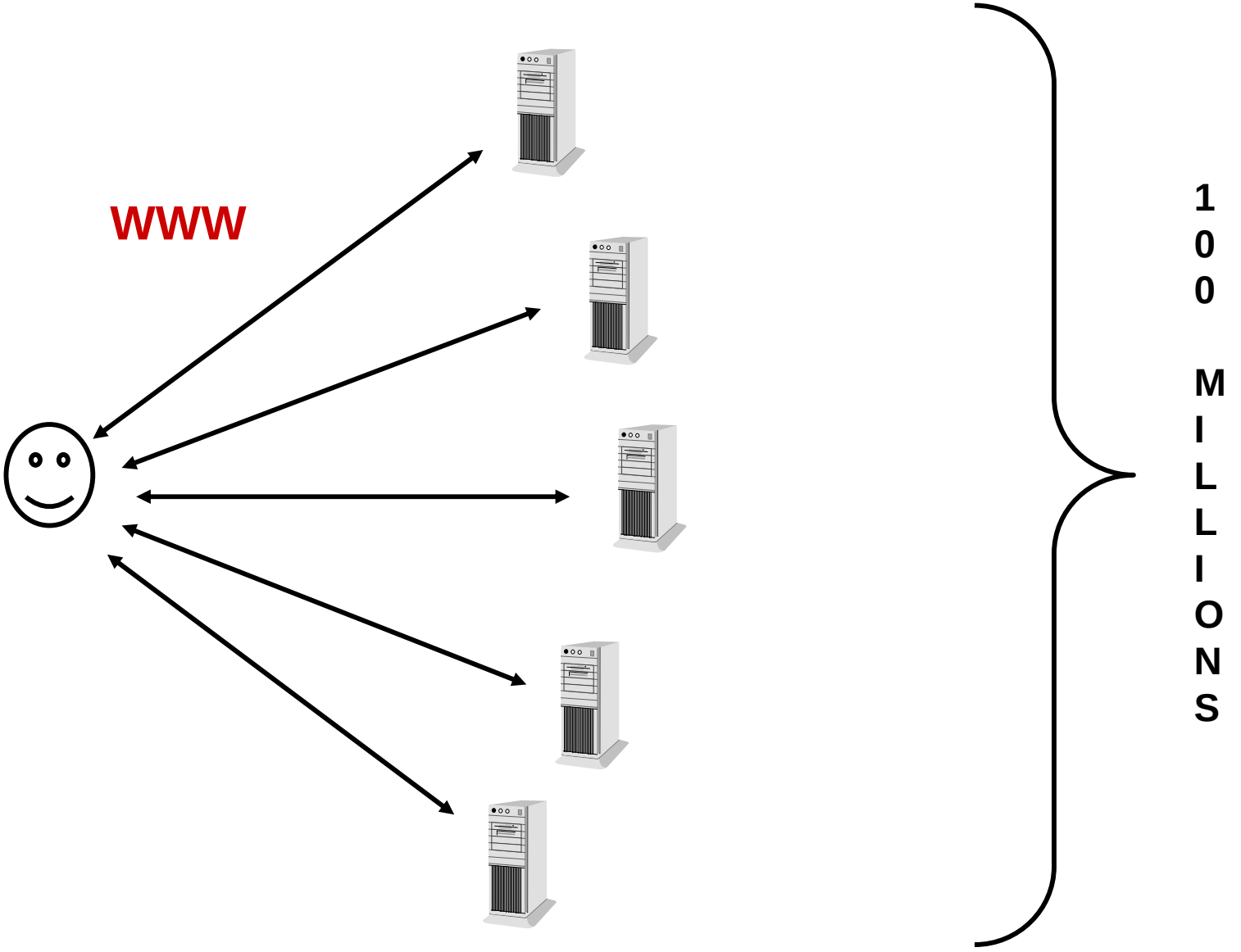
Internet 0: human <--> few humans, radio, TV



Internet 1: human <--> many people

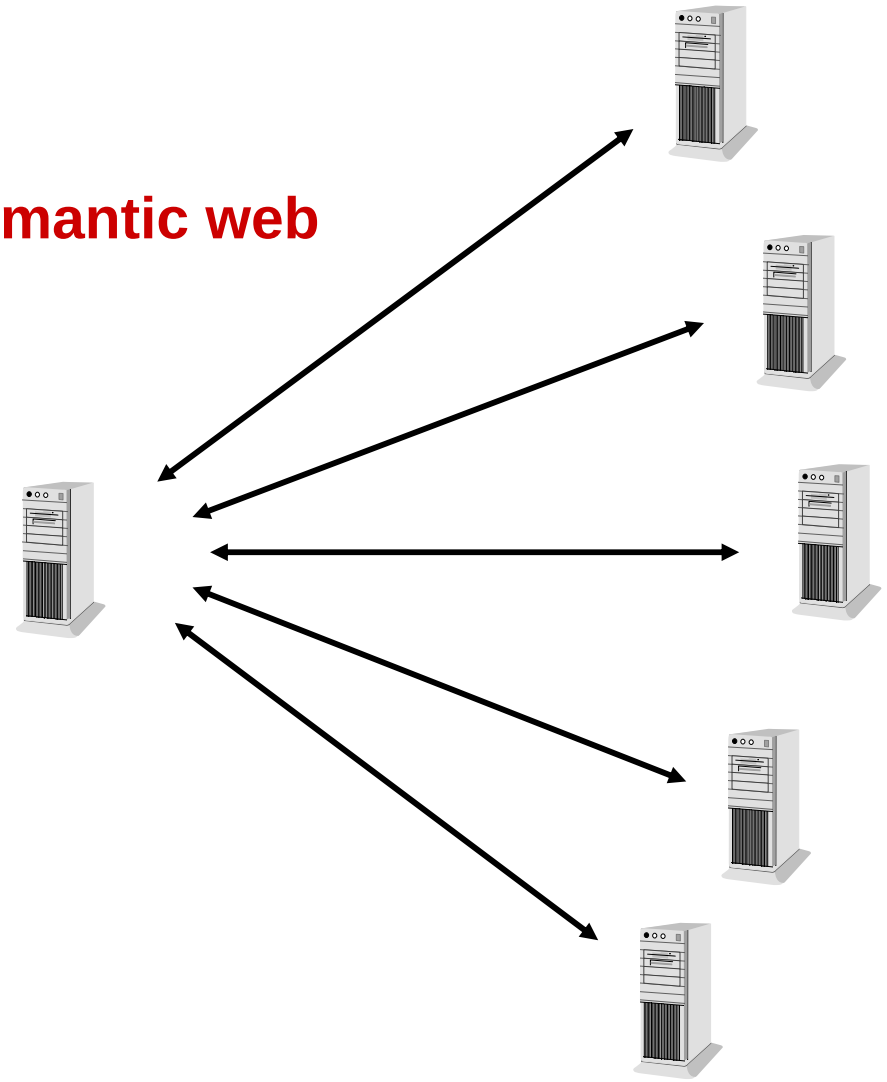


Internet 2: human <--> many computers



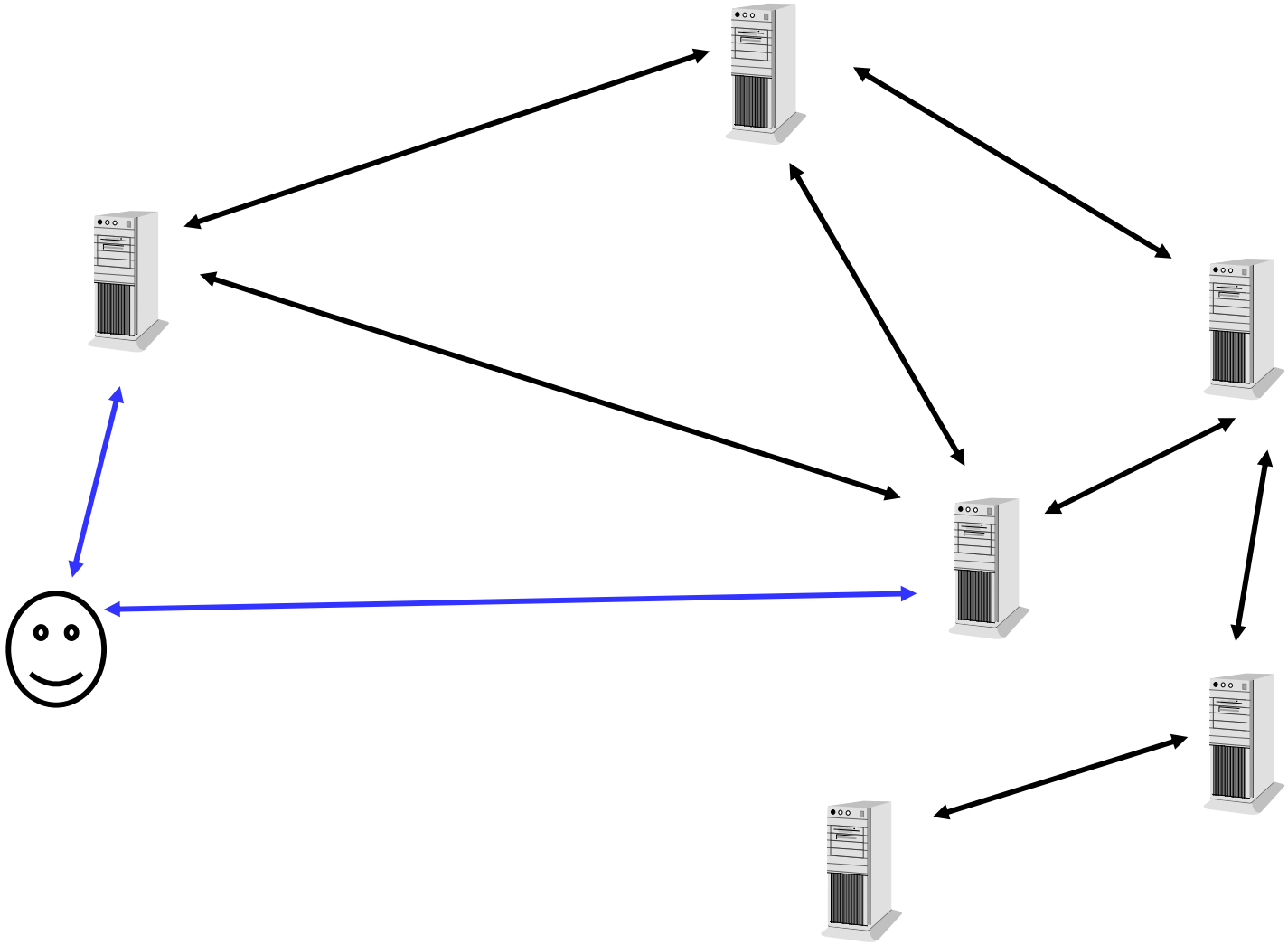
Internet 3: computer <--> many computers

XML + Semantic web

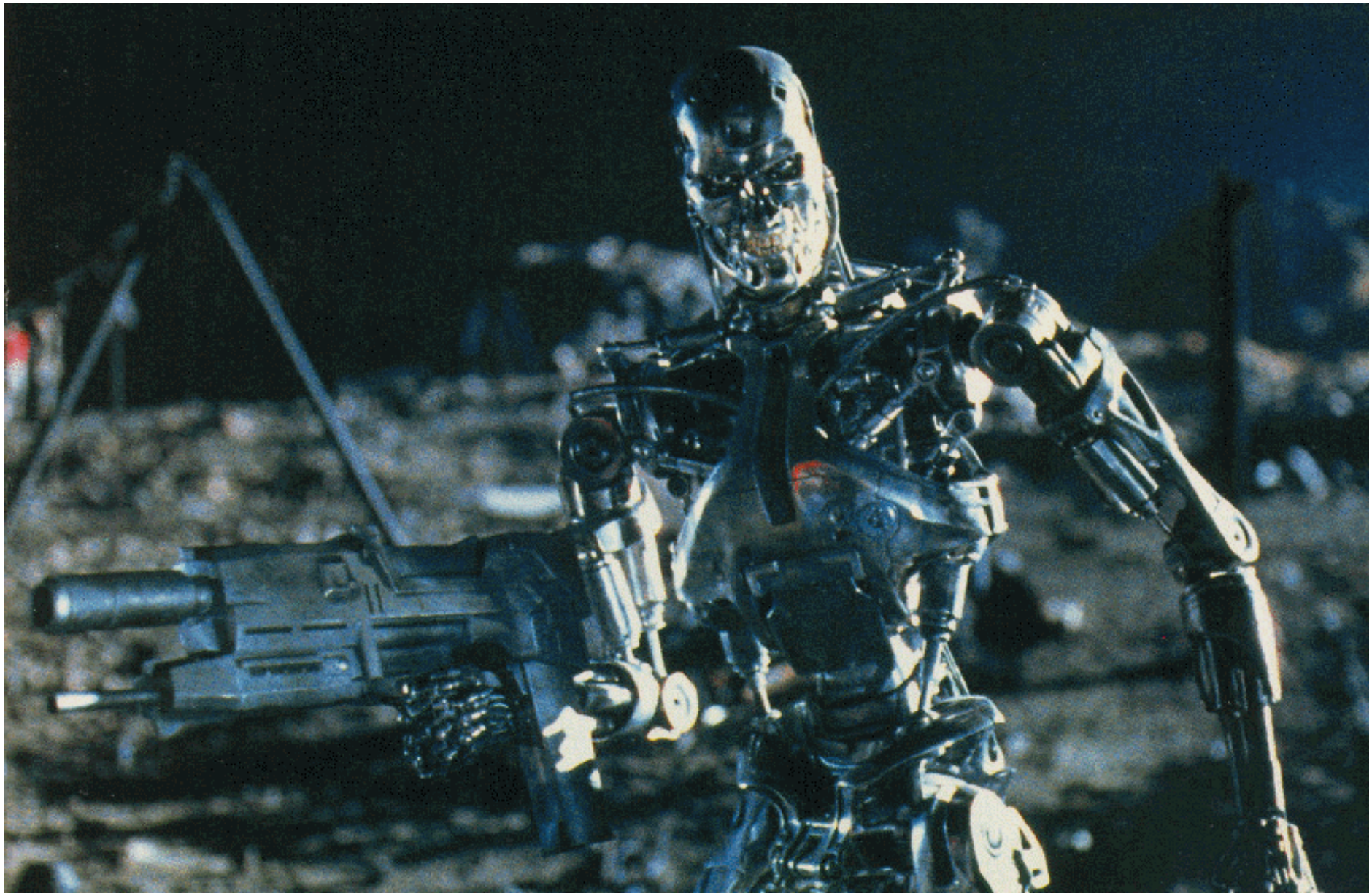


1 0 0 M I L L I O N S

Internet 3: human ??



Internet 4: Skynet?



HTML and XML

- **Info is put between “tag”s**
- **HTML: “tag” has visual semantics**

Here is bold text

Here is text in italics

- **XML: “tag” has no semantics**

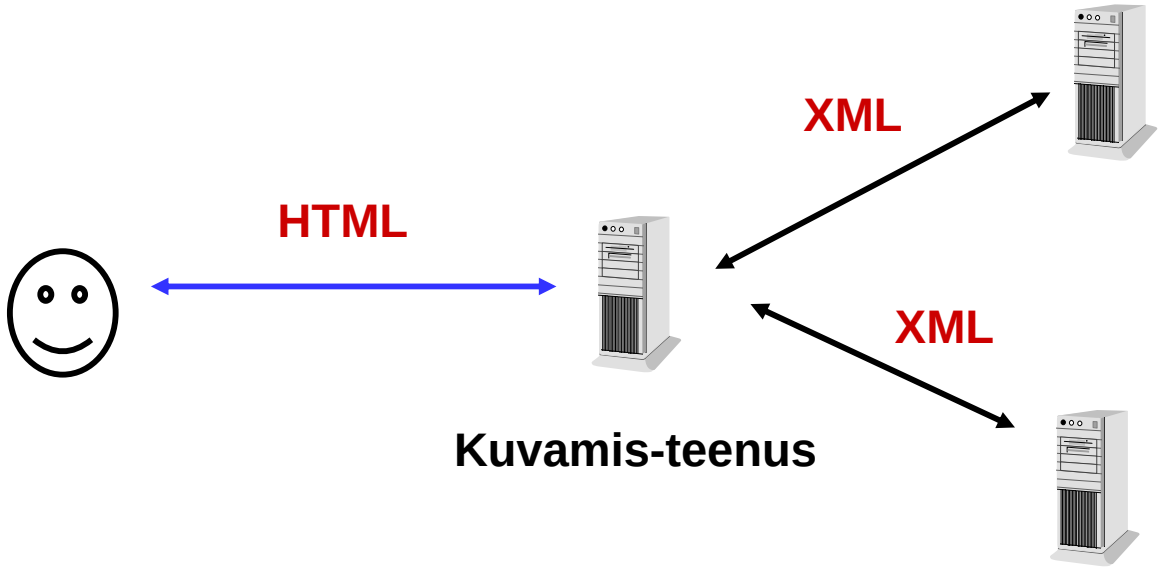
<author>**An Author**</author>

<myadress>**Astreet 5**</myadress>

Network services

- **Problem: very difficult to make a program that reads an HTML-page from another server.**

- **Idea: we make a network page in XML so that a program in another computer can read it**



Network services and .NET

- **Ballmer:**

  ".NET is at the core of what we're trying to do as a company.“

- **.NET: platform for creating network services**
  - Programming platform (C#, CLR, …)
  - XML-applications in XML language SOAP, WSDL, UDDI
  - Server support for applications

- **SOAP:**
  - XML-RPC further development: XML-communication language over http
  - TOETUS: MSFT, IBM, SUN, …

Applications ...

- **Intelligent search from the network**
- **Business-to-business solutions:**
  - Wholesales put their prices on the net, retailer search deal with proper price and conditions
  - Connecting different info-systems (2 banks merge, data has to be merged, there are many small systems in a bank)
- **Meeting planning: finding common times**
- **Access right system: who can access what catalog/information**
- **Intelligent assistant: searches itself information, prices I'm interested in; tells if something is found**

Procedural vs declarative

- **Procedural info presentation:**

  info is coded into the program.

- **Declarative info presentation:**

  info as separate rules (in files), which is built by an algorithm that uses algorithm-rules

What's missing ?

- **One program should understand the XML CONTENT given by another program**
- **Beer-shipment searching server:**

Wholesale A gives info:

```
<beer>
  <name>Guinness</name>
  <price>100</price>
</beer>
```

Wholesale b gives info:

```
<olu>
  <mark>Guinness</mark>
  <hind>100</hind>
</olu>
```

Wholesale C gives info:

```
<porter>
  <name>Guinness</name>
  <price>100</price>
</porter>
```

Differing languages

- We should know that:

BEER = ÕLU
PORTER is also BEER

- There is no hope to “prepare for” all languages
- There are sub-languages (general product language, beer language in Estonia, Czech, China, etc.)
- Language has to be interpreted by ourselves
- To translate you need rules and elementary understanding

Same problem in XML database in ONE MACHINE

- **we have tables in SQL language:**

ID	Name
1	Jaan Tamm
2	Ants Raud
3	Jaan Tamm

PID	Profession
1	Taster
2	Doorman
1	Salesman
3	Director

- **Rules:**
  - ID's in the first table are not allowed to repeat
  - PID in the second table has to exist in the first table

- **Needed: connection of names, rules for structures**